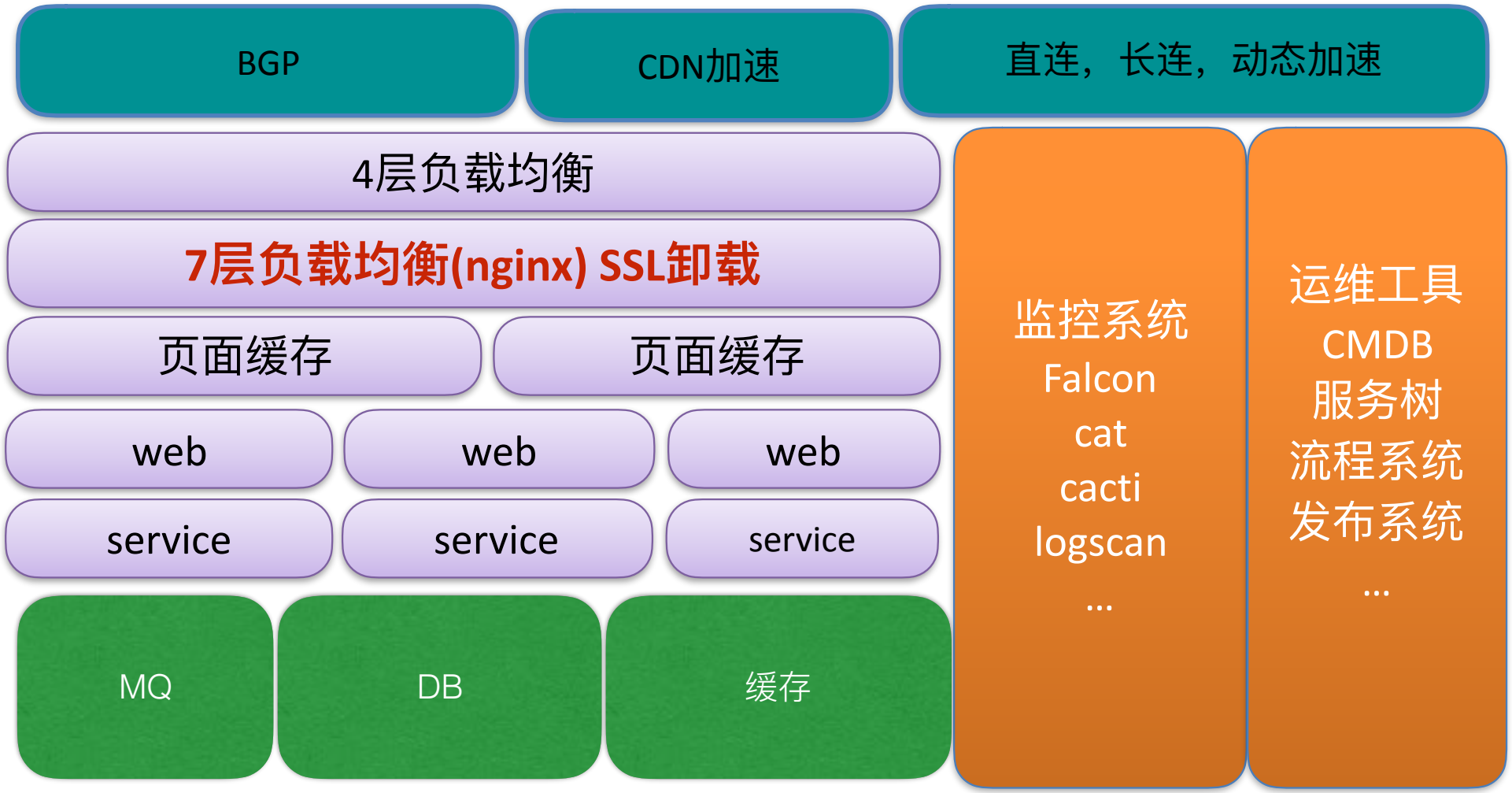
美大Nginx管理实践

许奎@运营开发团队

我是谁?

- 许奎
- 2014年加入美大
- 2014年 - 2015年 业务运维
- 2015年 - 至今 运维开发
- 目前上海运维开发团队负责人


BGP
CDN加速
直连，长连，动态加速
4层负载均衡
7层负载均衡(nginx) SSL卸载
页面缓存
页面缓存
web
web
web
service
service
service
MQ
DB
缓存
监控系统
Falcon
cat
cacti
logscan
...
运维工具
CMDB
服务树
流程系统
发布系统
...

状态

- K级别站点数量
- 百级别Nginx数量
- 日站点规则变更量50+
- 日Upstream变更量5000+

问题&痛点

- 效率低下
- 通过SVN或Git，多人协作，人肉维护配置文件
- 编写风格差异太大，质量层次不齐
- 没有结构化，没有api，无法接入自动化体系

非常危险

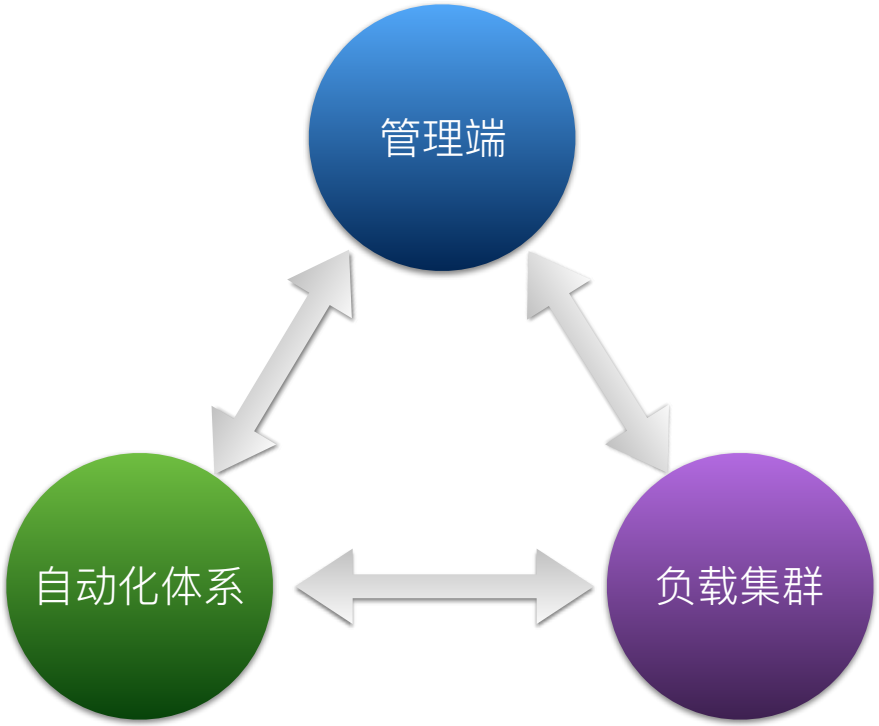
- 无有效验证手段
- 一个站点编写错误，可能导致整个集群异常
- 小问题居高不下

项目目标

- 实现web化管理
- 抽象结构化配置文件
- 丰富api，接入自动化体系，和其他系统有效结合
- 精细的权限控制，完整变更记录

分析

- 站点配置管理
- 负载集群管理
- 接入自动化体系



抽象配置

```
limit_conn_zone $server_name zone=perservers:20m;
limit_req_zone $host zone=500_limit:140m rate=500r/s;

server {
    server_name [redacted]

    ## log by lua for nginx monito";
    set $xlocation "[redacted]";
    log_by_lua_file /etc[redacted]_log.lua;

    proxy_connect_timeout 600ms;
    proxy_next_upstream off;
    access_log [redacted]log.nginx main buffer=64k flush=30s;

    gzip_comp_level 5;
    gzip_types application/json;

    location / {
        set $xlocation "mobile_log.nginx"; # must be first directive in location block
    error_page   500 502 503 504 = @fallback_report;
        proxy_pass http://data_mobile/;
    }
    location /data {
        set $xlocation "mobile_log.nginx"; # must be first directive in location block

        access_log off;
        error_page   500 502 503 504 = @fallback_report;
        proxy_connect_timeout 600ms;
        proxy_next_upstream off;
        rewrite /data/ms /ms break;
        rewrite /data/(.*)$ /$1 break;
        add_header "Access-Control-Allow-Origin" $allow_origin;
        #limit_req_status 480;
        #limit_req zone=500_limit burst=10;

        proxy_pass http://data_mobile;
    }
    location @fallback_report {
            return 200;
    }
}
```

- 结构化配置
  - 站点属性
  - 路由
  - 指令
  - Pool(Upstream)

DB

站点管理 - 属性

- 定义站点属性
  - 所属业务线
  - 端口
  - 域名
  - 所在集群（nginx集群）
  - 开启https
  - 证书
  - 开启日志
  - 开启监控

- 设定规范

- 简单开关

站点结构化配置-Location

- 映射规则（Location配置），**最依赖的功能之一**
  - 匹配RUL进入对于的Location
  - 执行Location包含的指令
  - 根据指令，转发请求到后端服务

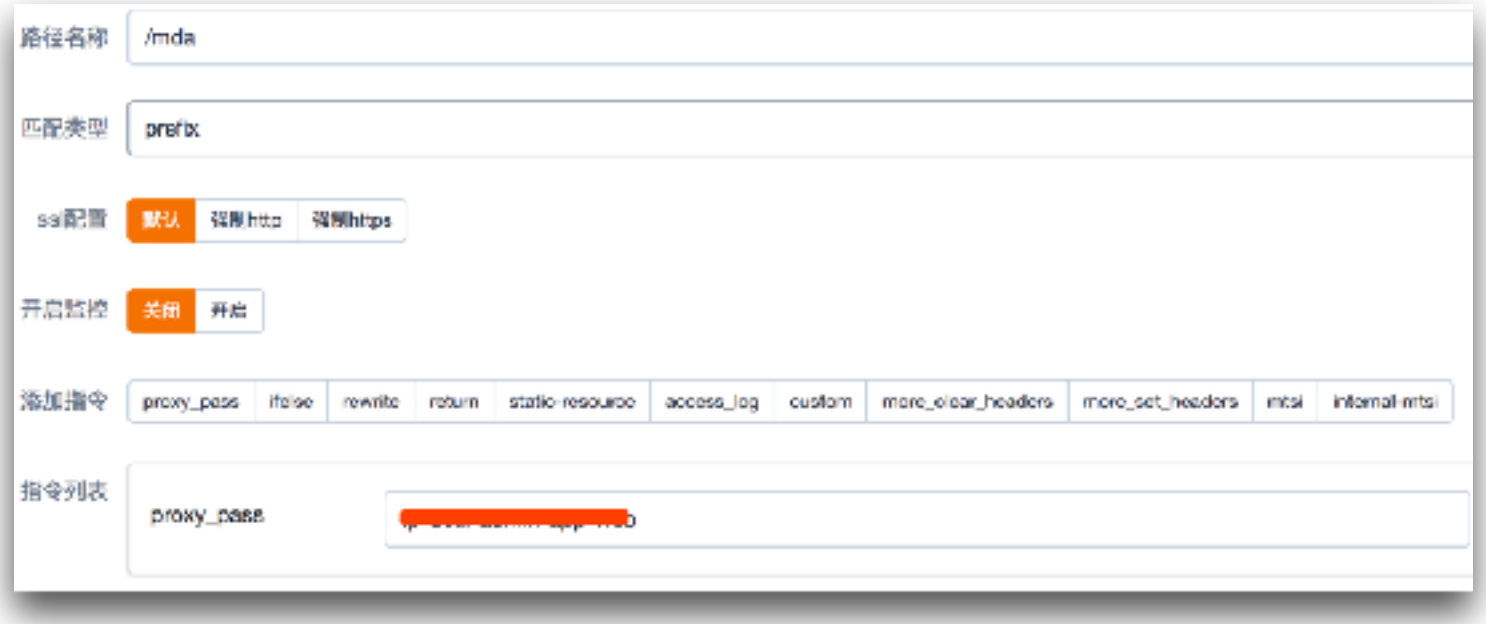
Location-存储

- Location独立一张表
- 每一个Location一条记录
- 使用双向链表，保证Location顺序
- 通过ID，与站点关联

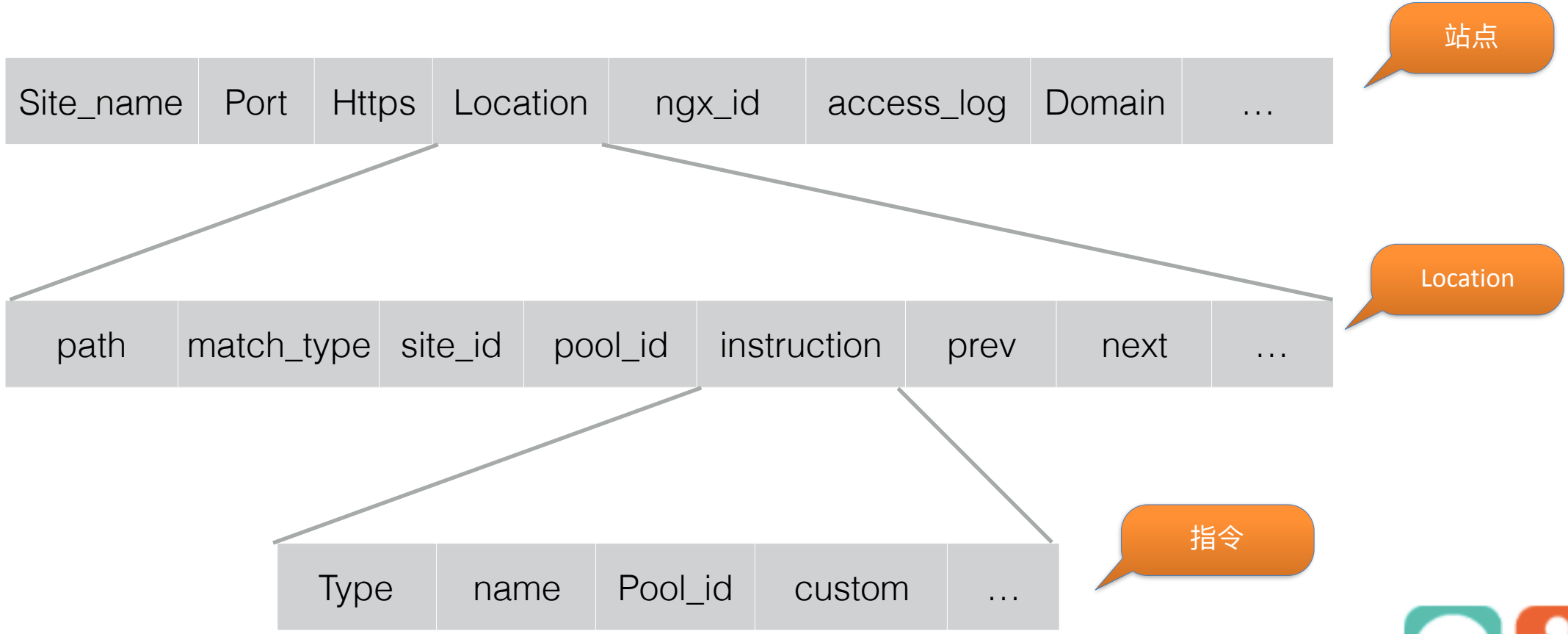
path	match_type	site_id	pool_id	instruction	prev	next	…

Location-指令

- 封装Location下面常用指令（官方和自定义）
  - proxy_pass
  - rewrite
  - return
  - access_log
  - if else
  - mtsi（自定义指令）



站点-Location包含关系

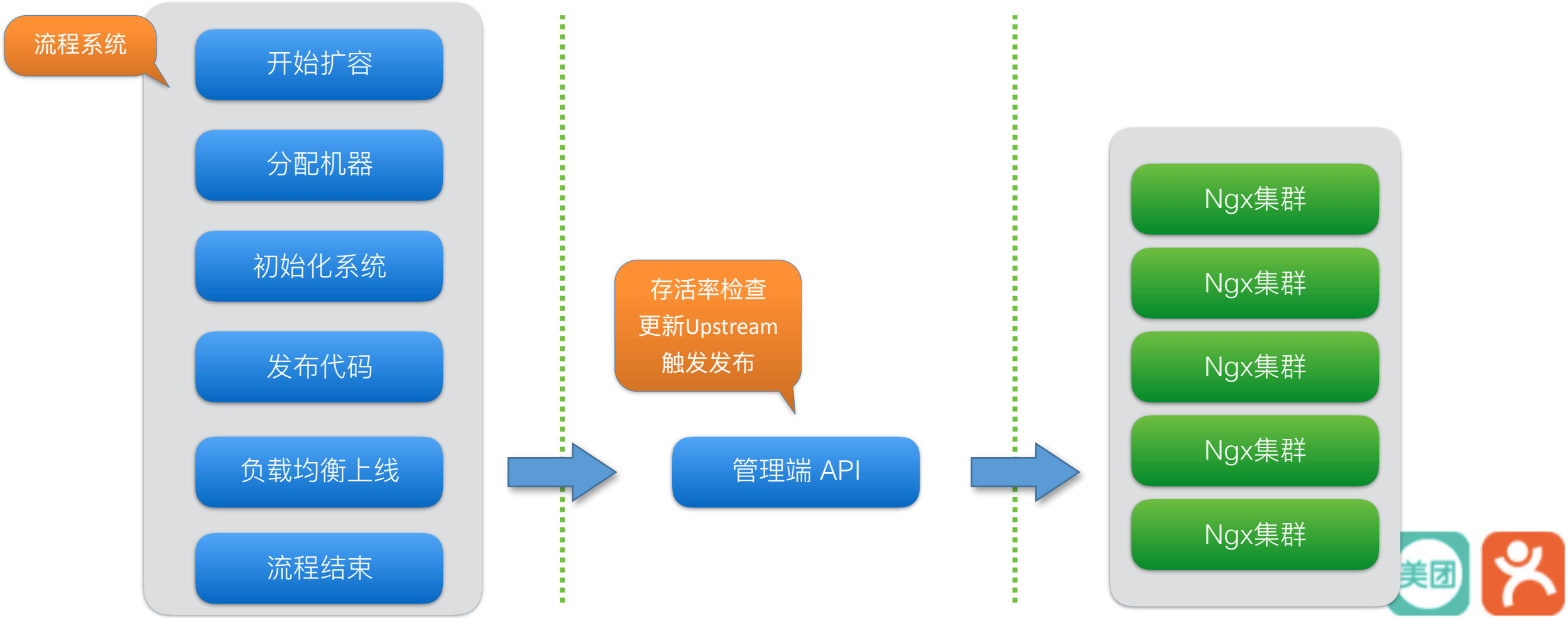


站点结构化配置-Pool

- Pool（Upstream）管理
  - 强制绑定CMDB项目
  - 同步节点(member)增删
  - 原子化


全部Pool
配置信息
节点管理
相关站点
自定义发布
发布站点
删除Pool
编辑
Name IP 端口 权重 最大失败次数 失败超时时间 状态
1 80 1 3 2s ENABLED
2 80 1 3 2s ENABLED
3 80 1 3 2s ENABLED
4 80 1 3 2s ENABLED
5 80 1 3 2s ENABLED
6 80 1 3 2s ENABLED
7 80 1 3 2s ENABLED

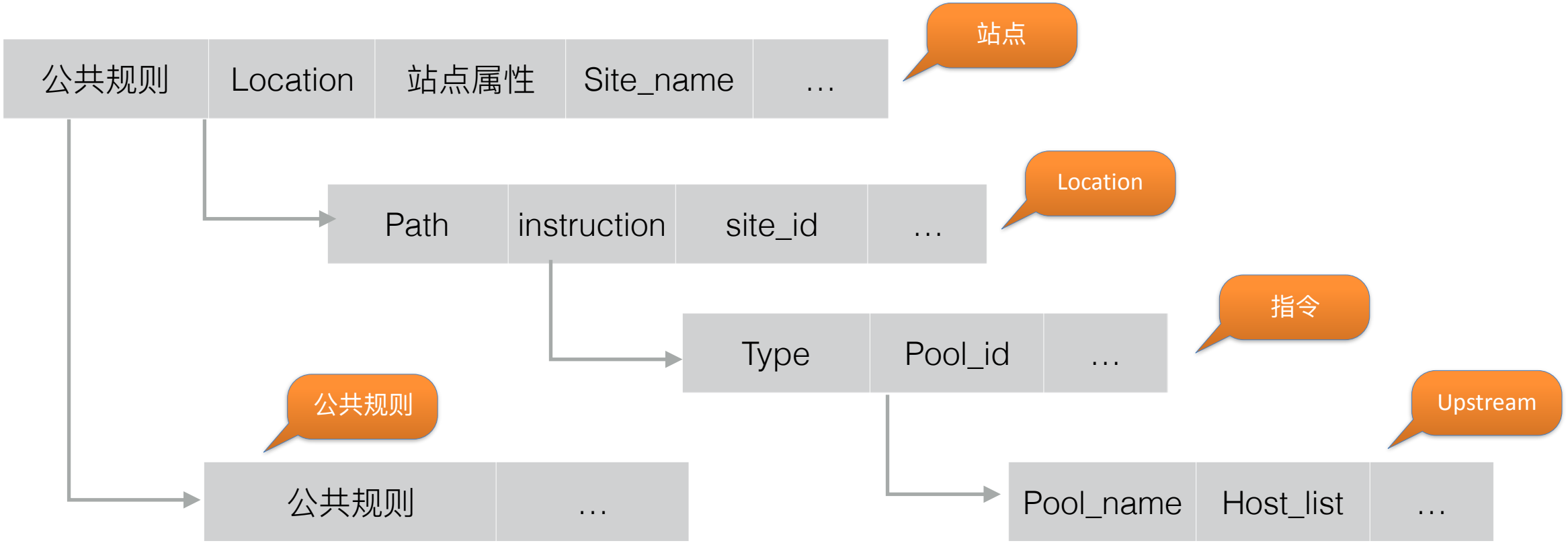
Pool增删


流程系统
开始扩容
分配机器
初始化系统
发布代码
负载均衡上线
流程结束
存活率检查
更新Upstream
触发发布
管理端 API
Ngx集群
Ngx集群
Ngx集群
Ngx集群
Ngx集群

结构化配置-公共规则

- 公共规则
  - 必须预先定义
  - 全局性
  - 可复用

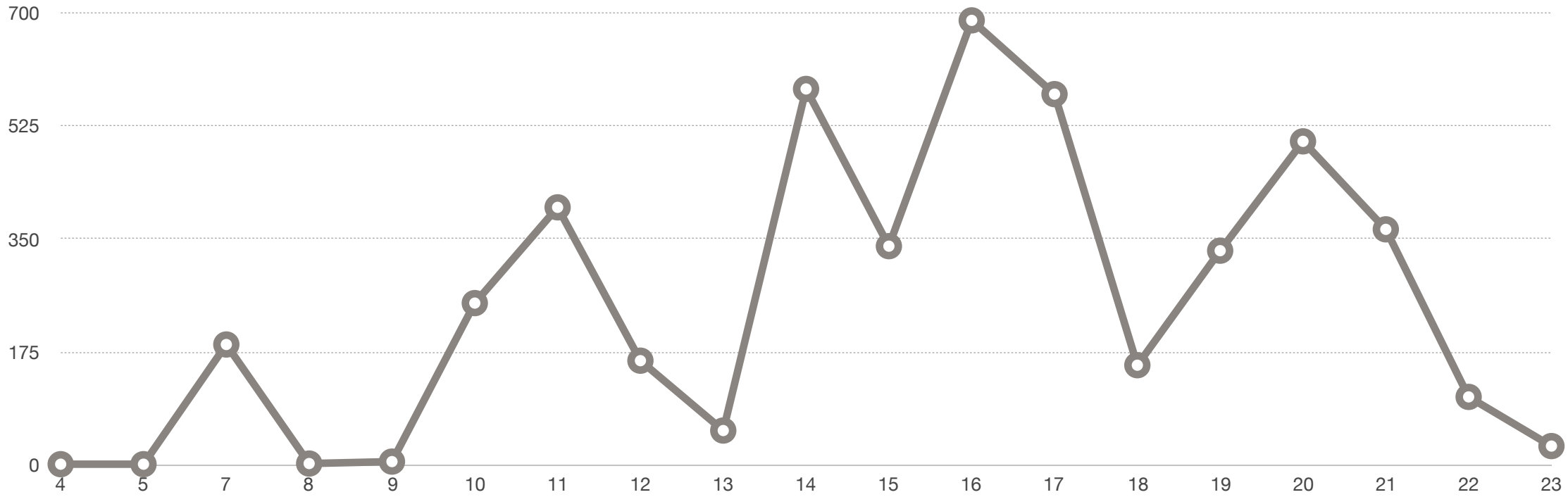
完整站点


公共规则
Location
站点属性
Site_name
…
站点
Path
instruction
site_id
…
Location
Type
Pool_id
…
指令
公共规则
公共规则
…
Upstream
Pool_name
Host_list
…

集群管理

- 核心功能
  - 站点和集群映射绑定关系
  - 配置文件落地，即站点发布

日常发布量 5000+



发布-人工触发

- 触发条件
  - 站点配置变更
  - 映射规则变更
  - 证书变更
- 发布动作
  - 生成配置文件
  - 推送到集群所有集群
  - reload nginx

发布-自动触发（5000+）

- 触发条件（Upstream节点增删）
  - Pool变更
  - 服务的 扩容，缩容
  - 业务代码发布
- 发布动作
  - 生成配置文件
  - 推送到集群所有集群
  - dyups内存接口更新 Upstream member

发布到底干了什么

理论上

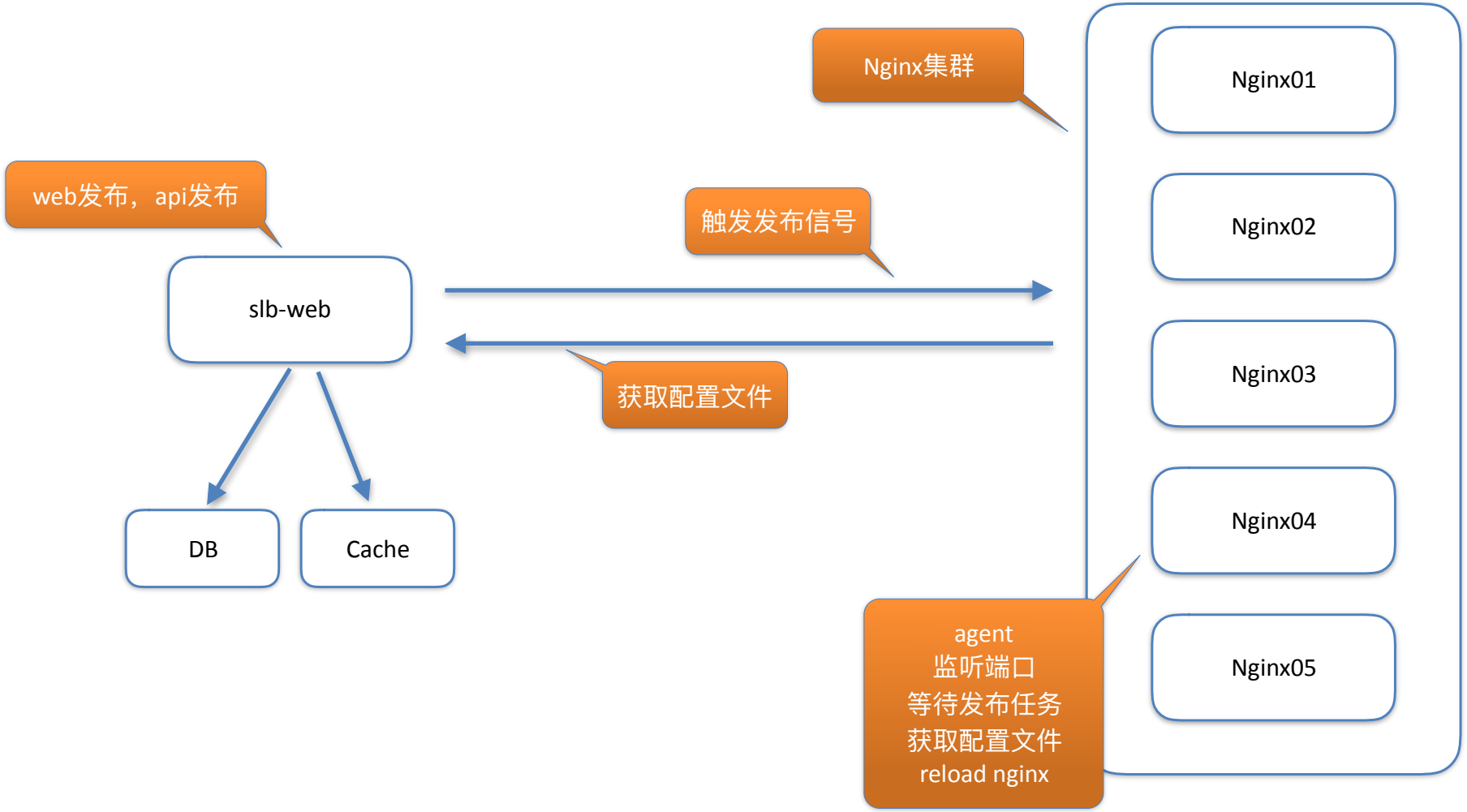
1. 查询DB数据，渲染配置文件
2. 拷贝到Nginx集群
3. Reload Nginx 让其配置生效

实际情况总是复杂一些：

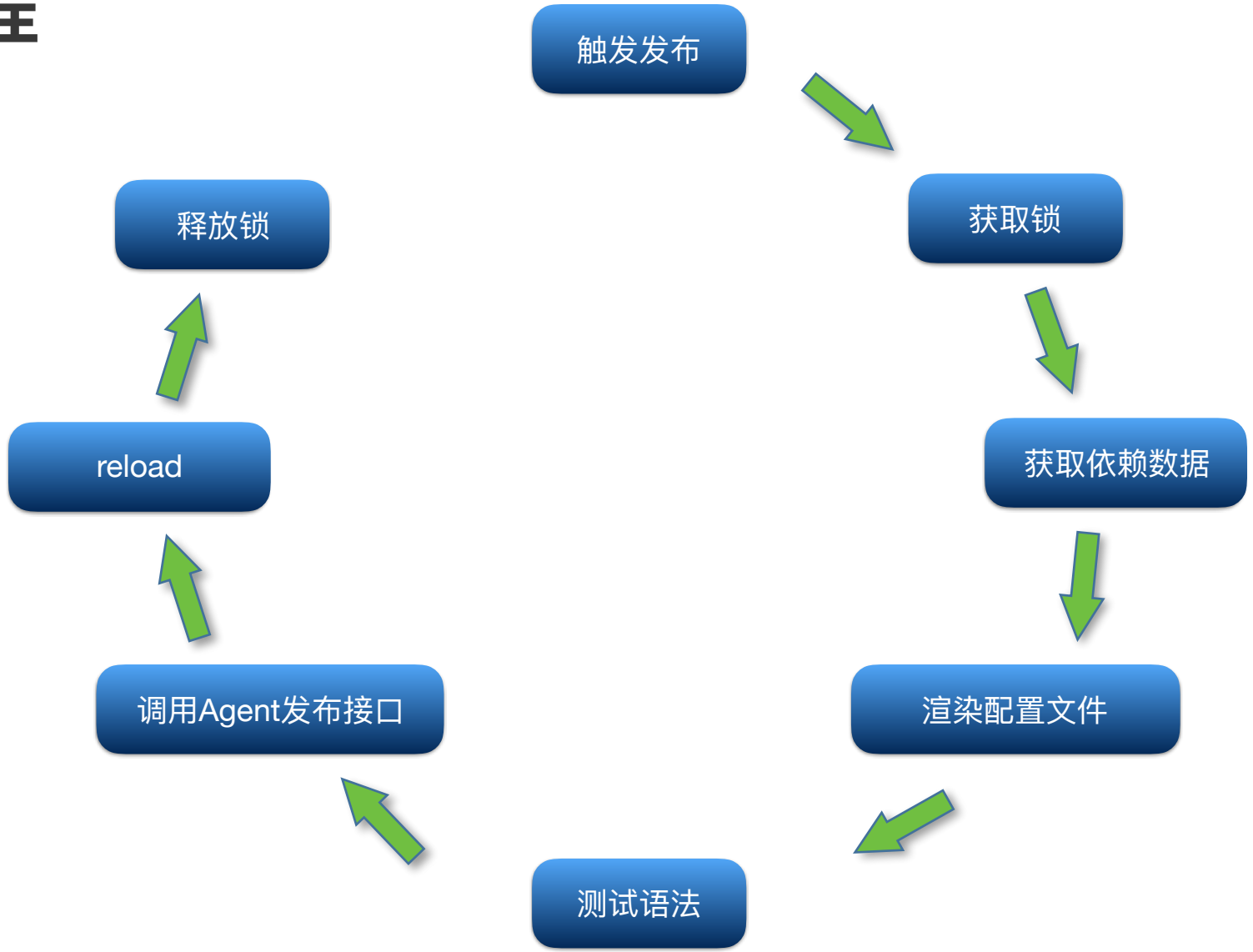
1. 监测磁盘空间
2. 语法问题
3. 发布平率可能非常高，并发reload
4. reload有损
5. upstream存活比例
6. 部分成功
7. 发布速度

…

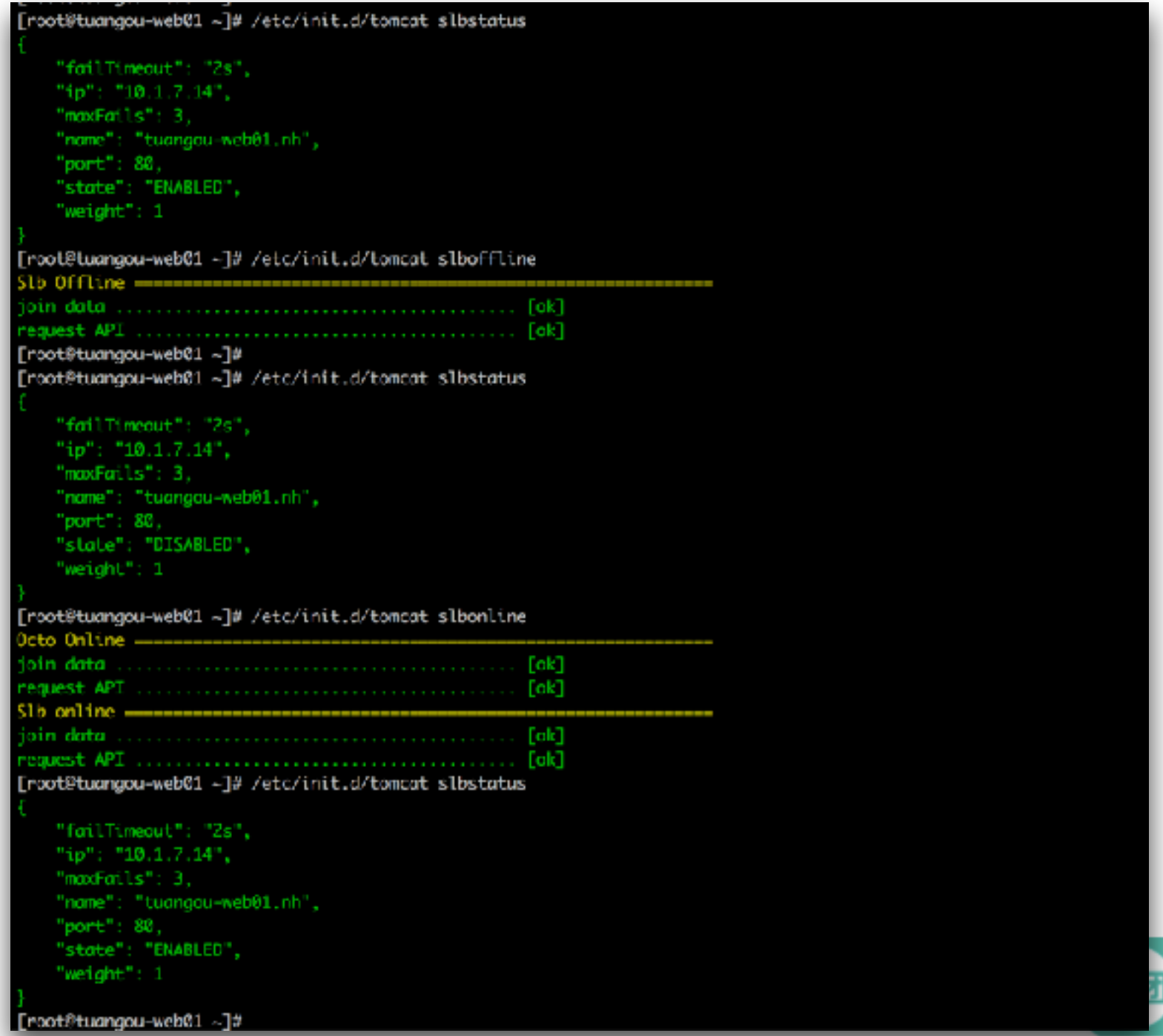
发布架构


Nginx集群
Nginx01
Nginx02
Nginx03
Nginx04
Nginx05
web发布，api发布
slb-web
触发发布信号
获取配置文件
DB
Cache
agent
监听端口
等待发布任务
获取配置文件
reload nginx

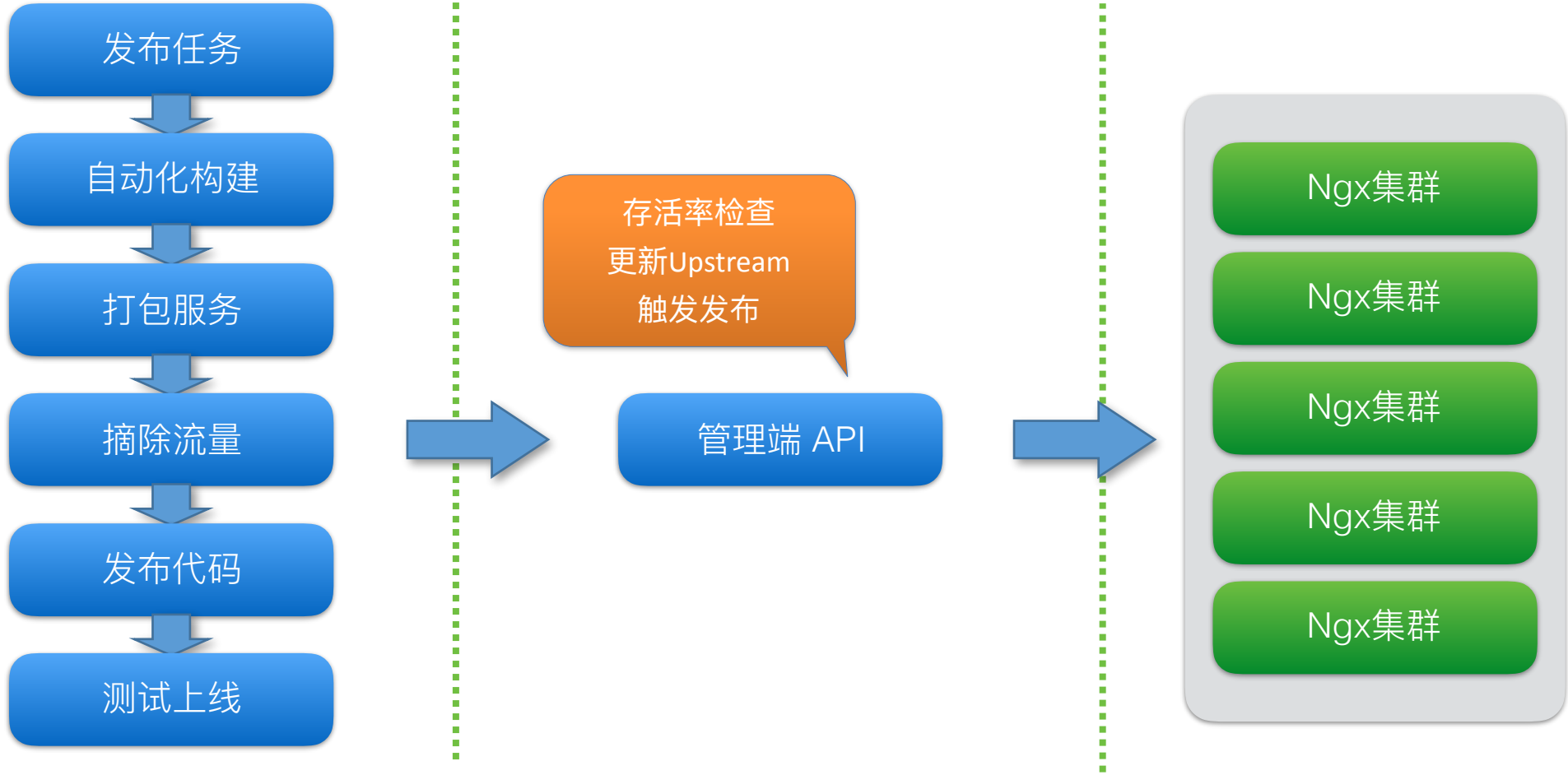
发布过程



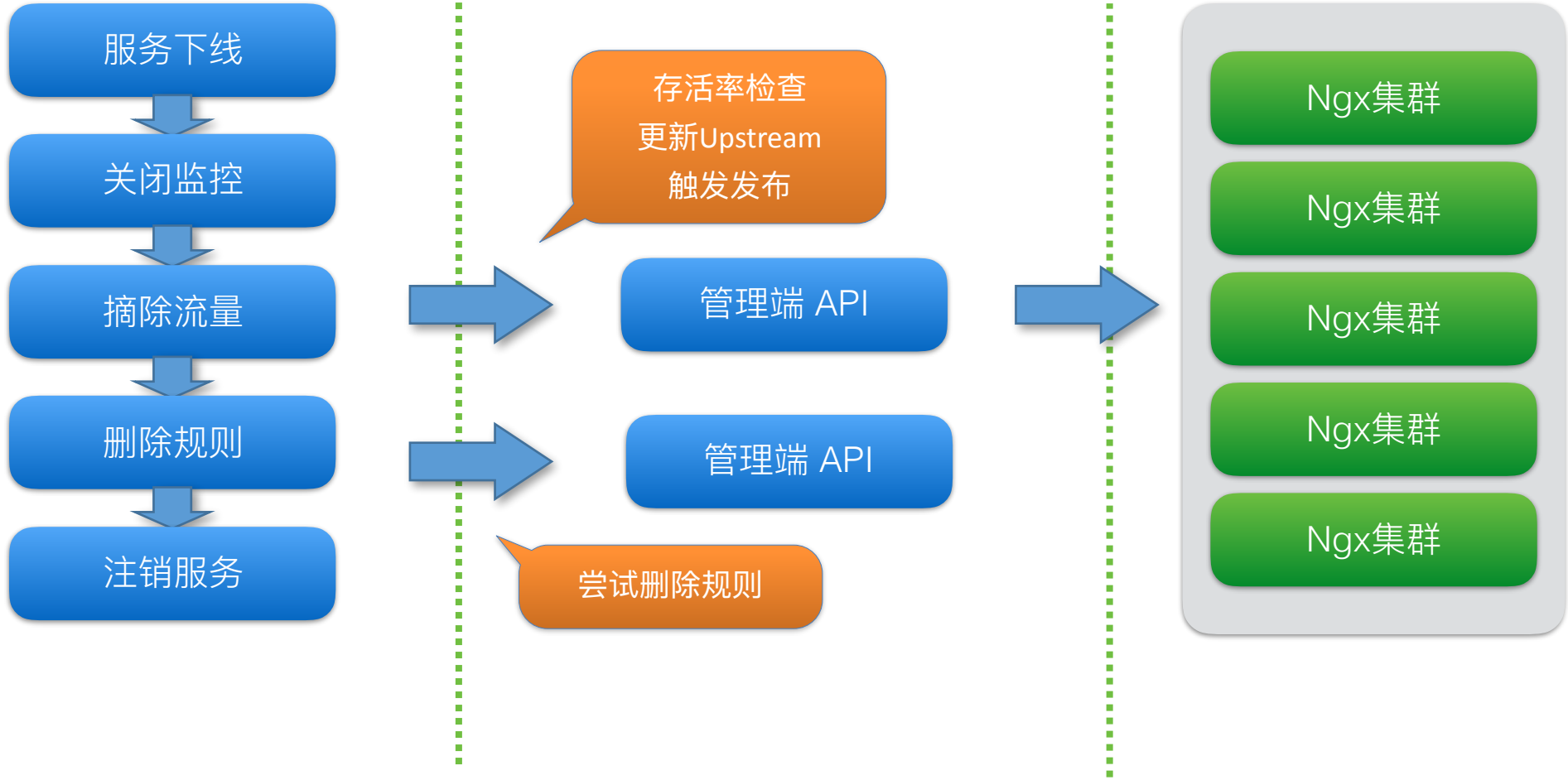
- 一次api发布范例
  - 查看状态：enable
  - 操作下线：slboffline
  - 查看状态：disable
  - 操作上线：slbonline
  - 查看状态：enable



接入自动化体系


发布任务
自动化构建
打包服务
摘除流量
发布代码
测试上线
存活率检查
更新Upstream
触发发布
管理端 API
Ngx集群
Ngx集群
Ngx集群
Ngx集群
Ngx集群

接入自动化体系


服务下线
关闭监控
摘除流量
删除规则
注销服务
存活率检查
更新Upstream
触发发布
管理端 API
管理端 API
尝试删除规则
Ngx集群
Ngx集群
Ngx集群
Ngx集群
Ngx集群

价值

- 统一web管理方式
- 精确权限控制，完整变更记录
- 统一渲染模板
- 完整测试，杜绝错误配置上线
- 完整丰富API

Q&A

谢谢